

```

1  /*
2  Display.cpp - A simple track GPS to SD card logger. Display module.
3  TinyTrackGPS v0.9
4
5  Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
6  All rights reserved.
7
8  rafael.reyes.carmona@gmail.com
9
10 This file is part of TinyTrackGPS.
11
12 TinyTrackGPS is free software: you can redistribute it and/or modify
13 it under the terms of the GNU General Public License as published by
14 the Free Software Foundation, either version 3 of the License, or
15 (at your option) any later version.
16
17 TinyTrackGPS is distributed in the hope that it will be useful,
18 but WITHOUT ANY WARRANTY; without even the implied warranty of
19 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
20 GNU General Public License for more details.
21
22 You should have received a copy of the GNU General Public License
23 along with TinyTrackGPS. If not, see <https://www.gnu.org/licenses/>.
24 */
25
26 #include "Display.h"
27
28 Display::Display(Display_Type t):_screen(t){
29     //if (_screen == SDD1306_128X64){
30         _width = 16;
31         _height = (_screen > 0) ? 2 : 8;
32         //_offset = 0;
33     //} else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
34         //    _width = 16;
35         //    _height = 2;
36         //_offset = 0;
37     //}
38 }
39
40 void Display::start(){
41     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
42         #if defined(DISPLAY_TYPE_LCD_16X2)
43         lcd = new LiquidCrystal(RS, ENABLE, D0, D1, D2, D3);
44         lcd->begin(_width, _height);
45         #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
46         lcd = new LiquidCrystal_I2C(I2C,_width,_height);
47         lcd->init();
48         lcd->backlight();
49         #endif
50
51         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
52         // DEFINICION DE CARACTERES PERSONALIZADOS
53         static byte alt[8] = { 0x04, 0x0E, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04 };
54         static byte ant[8] = { 0x0E, 0x11, 0x15, 0x11, 0x04, 0x04, 0x0E, 0x00 };
55         static byte sd[8] = { 0x0E, 0x11, 0x1F, 0x00, 0x00, 0x17, 0x15, 0x1D };
56         static byte hourglass_0[8] = { 0x1F, 0x0E, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
57         0x1F };
58         static byte hourglass_1[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0A,
59         0x1F };

```

```

58     static byte hourglass_2[8] = { 0x1F, 0x0A, 0x0E, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
59     static byte hourglass_3[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0A, 0x0E,
0x1F };
60     static byte hourglass_4[8] = { 0x1F, 0x0A, 0x0A, 0x04, 0x04, 0x0E, 0x0E,
0x1F };
61     lcd->createChar(0, hourglass_0);
62     lcd->createChar(1, hourglass_1);
63     lcd->createChar(2, hourglass_2);
64     lcd->createChar(3, hourglass_3);
65     lcd->createChar(4, hourglass_4);
66     lcd->createChar(5, alt);
67     lcd->createChar(6, ant);
68     lcd->createChar(7, sd);
69     #endif
70     //}
71
72     //if (_screen == SDD1306_128X64) {
73         #if defined(DISPLAY_TYPE_SDD1306_128X64)
74         u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
75         u8x8_SSD1306->begin();
76         u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
77         #endif
78     //}
79 }
80
81 void Display::clr(){
82     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
83         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
84         lcd->clear();
85         #endif
86     //}
87     //else if (_screen == SDD1306_128X64) {
88         #if defined(DISPLAY_TYPE_SDD1306_128X64)
89         u8x8_SSD1306->clear();
90         #endif
91     //}
92 }
93
94 void Display::print(int vertical, int horizontal, const char text[]){
95     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
96         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
97         lcd->setCursor(vertical, horizontal);
98         //lcd->print(text);
99         #endif
100     //}
101     //else if (_screen == SDD1306_128X64) {
102         #if defined(DISPLAY_TYPE_SDD1306_128X64)
103         //u8x8_SSD1306->setCursor(vertical, (horizontal*2));
104         //u8x8_SSD1306->print(text);
105         u8x8_SSD1306->setCursor(vertical, (horizontal*2));
106         //this->print(text);
107         //u8x8_SSD1306->display();
108         #endif
109     //}
110     this->print(text);
111 }
112
113 void Display::print(int line, const char text[]){

```

```

114     byte pos = _width -(strlen(text));
115     pos = (pos >> 1);
116     this->print((int)pos, line, text);
117 }
118
119 void Display::print(const char text[]){
120     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
121     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
122     lcd->print(text);
123     #endif
124     //}
125     //else if (_screen == SDD1306_128X64) {
126     #if defined(DISPLAY_TYPE_SDD1306_128X64)
127     u8x8_SSD1306->print(text);
128     u8x8_SSD1306->flush();
129     #endif
130     //}
131 }
132
133 void Display::print(const char text1[], const char text2[]){
134     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
135     this->print((_screen > 0)?0:1, text1);
136     this->print((_screen > 0)?1:2, text2);
137     //}
138     //else if (_screen == SDD1306_128X64) {
139     //    this->print(1, text1);
140     //    this->print(2, text2);
141     //}
142 }
143
144 void Display::print(const char text1[], const char text2[], const char text3[]){
145     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
146     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
147     this->print(text1, text2);
148     delay(750);
149     //for (unsigned long start = millis(); millis() - start < 750;) {}
150     //unsigned long start = millis();
151     //do {} while (millis() - start < 750);
152
153     this->clr();
154     this->print(0,text3);
155     //}
156     #endif
157     #if defined(DISPLAY_TYPE_SDD1306_128X64)
158     //else if (_screen == SDD1306_128X64) {
159     this->print(0, text1);
160     this->print(1, text2);
161     this->print(2, text3);
162     //}
163     #endif
164 }
165
166 void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){
167     #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
168     this->print(text1,text2);
169     delay(750);
170     this->print(text3,text4);
171     #endif
172     #if defined(DISPLAY_TYPE_SDD1306_128X64)

```

```

173     this->print(0, text1);
174     this->print(1, text2);
175     this->print(2, text3);
176     this->print(3, text4);
177     #endif
178 }
179
180 void Display::wait_anin(unsigned int t){
181     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
182         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
183             lcd->setCursor(15,1);
184             lcd->write((byte)t%5);
185             #endif
186         //}
187     //else if (_screen == SDD1306_128X64) {
188         #if defined(DISPLAY_TYPE_SDD1306_128X64)
189
190             const char p[4] = {(char)47,(char)45,(char)92,(char)124};
191             u8x8_SSD1306->setCursor((_width-1),6);
192             u8x8_SSD1306->print(p[t%4]);
193
194             /*
195             static uint8_t hourglass_UP[5][8] = {
196                 0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
197                 0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
198                 0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
199                 0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
200                 0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
201             };
202
203             static uint8_t hourglass_DOWN[5][8] =
204             {0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
205                 0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
206                 0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
207                 0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
208                 0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
209             };
210
211             u8x8_SSD1306->drawTile((_width-1), 6, 1, hourglass_UP[t%5]);
212             u8x8_SSD1306->drawTile((_width-1), 7, 1, hourglass_DOWN[t%5]);
213             */
214             #endif
215         //}
216     }
217
218 void Display::print_PChar(byte c) {
219     //if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
220         #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
221             lcd->write(c);
222             #endif
223         //}
224     //else if (_screen == SDD1306_128X64) {
225         #if defined(DISPLAY_TYPE_SDD1306_128X64)
226
227             static uint8_t PChar_UP[3][8] = { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
228                 0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
229                 0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
230             };
231
232             static uint8_t PChar_DOWN[3][8] = { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
233                 0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
234                 0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8

```

```

231         };
232
233         //char tile;
234         if (c == 5) {
235             //tile = (char)0x18;
236             u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
237             u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
238             //u8x8_SSD1306->setCursor(9, 2);
239         }
240         else if (c == 6) {
241             //tile = (char)0x7f;
242             u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
243             u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
244             //u8x8_SSD1306->setCursor(11, 0);
245         }
246         else if (c == 7) {
247             //tile = (char)0xda;
248             u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
249             u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
250             //u8x8_SSD1306->setCursor(15, 0);
251         }
252         //u8x8_SSD1306->print(tile);
253         #endif
254         //}
255     }
256     /*
257     void Display::splash(int time_delay){
258         this->print(NAME, VERSION);
259         delay(time_delay);
260     }
261     */

```